Learning and performance of sequential action under different workload conditions: an ACT-R model

(unpublished long version of the poster presented at the 3rd ICCM 2000 in Groningen)

Wolfgang Schoppek

Human Factors & Applied Cognition

George Mason University

wschoppe@gmu.edu

Abstract: This paper describes an ACT-R model that learns and performs sequential actions and displays the skipping of steps under high workload. The model is part of a larger one that simulates the interaction between commercial airline pilots and the aircraft automation. Learning and performing scan patterns or procedures to program the flight management computer are examples for sequential actions in that domain. Errors occurring in the interaction with the automation often involve step-skipping, particularly under high cognitive workload. The model assumes no special learning strategy, but uses two basic ACT-R learning mechanisms, baselevel learning and associative learning. In the test phase, the sequences emerge by symbolically unconstrained retrievals of the respective next steps. The skipping of steps depends on the relative influences of baselevel values and associative activation which change with the varying workload. Relations to extant models of the serial recall paradigm and implications for training are discussed.

Introduction

The model described in this paper originates in a project on the abatement of automation errors in commercial aircraft (Boehm-Davis, Schoppek, Diez, Hansberger, Holt, & Ikomi 2000). Increased air traffic has made the use of advanced automation systems essential to obtain the precision required in the crowded airspace. But the enhanced precision of these systems seems to come at the cost of new types of errors that occur in the interaction between pilots and the autoflight systems (Sarter & Woods, 1992; 1995).

The main means for error reduction are improvements in training procedures and the design of automation devices. For both it is critical to understand the pilots' cognitive processes when they interact with the automation. Our approach combines cognitive task analysis with cognitive modeling of the pilots' behavior. The goal is to have a model that performs some important aspects of flying with the automation. Such a model will help to test assumptions about new ideas for error reduction much faster and cheaper compared to empirical testing alone, and more precisely than with verbal theories.

Much of the behavior of pilots when they use the automation can be characterized as sequential action (Irving, Polson, & Irving, 1994). This holds for the visual scan patterns that have to be learned as well as for programming the Flight Management Computer (FMC). Although the performance of sequential action is only a part of the final model, it is important to have an appropriate submodel of that aspect.

One type of error in the performance of sequences is the skipping of steps. Step-skipping can contribute to serious problems, e.g. incomplete scan patterns will result in an invalid representation of the situation, or a skipped step may lead to the FMC being misprogrammed. Step-skipping has been reported to be a particular problem in situations involving high cognitive workload (Sarter & Woods, 1992; Wiener, 1989; see also Wiegmann & Shappell, 1997, for a classification of human errors in 5008 aircraft accidents and incidents).

The question that shall be addressed in this paper is how the learning of sequential action can be modeled such that the model displays step-skipping under the condition of high cognitive workload. I developed the model with the example of learning and performance of a scan pattern (a certain sequence in which a number of displays is processed). But the model may apply to learning and performance of sequential action in general.

ACT-R model

We chose to build the model in ACT-R (Anderson & Lebiere, 1998), because of the advantages of using a unified cognitive architecture, as e.g. the broad empirical basis of the basic assumptions, or the comparability with other models. ACT-R 4.0 is a production system featuring declarative and procedural knowledge. The declarative memory elements - called chunks - are typed structures with slots that can be filled with other chunks, thus establishing a network of declarative knowledge. Production rules operate on declarative memory. ACT-R assumes a goal stack whose top element can be viewed as the focus of attention. The elements of the goal chunk are sources of activation which spreads into the network of declarative memory. The activation of a chunk reflects the probability that it is needed in the current context and determines its retrieval. Two learning mechanisms change this probability. One is baselevel learning which changes the baselevel value of a chunk as a function of its use history. The more frequently and the more recently a chunk has been used, the higher is its baselevel value. The second mechanism is associative learning. Associative weights (sji) between two chunks i and j are updated when chunk j is an element of the goal (and thus, a source of activation) while chunk i is retrieved from memory. The sji value depends, among other things, on how often chunk i is also retrieved in other contexts, and on the number of other contexts where chunk j is element of the goal while chunk i is not retrieved.

The task that was modeled begins with a supervised learning phase where the model is instructed to scan six displays, named A, B, C, D, E, and F, in alphabetical order. When the model asks for a place to look, a look command is provided, and the model follows that command. It recognizes the display, reads the displayed value, and memorizes the value. Then the model asks for the next place to look. In a test phase, the model performs scans without external hints.

One prerequisite for the model is that it has to store information about the sequence of steps. Two basic solutions for that are to establish symbolic links between consecutive steps, and to link them associatively. Symbolic linking results in non-ambiguous representations of given sequences, which makes it hard to model step-skipping without assumptions about how symbolic links are sometimes bypassed. There is no need for additional assumptions such as partial matching if associative links are used. Furthermore, as associative learning is already part of the ACT-R theory, it needs not to be modeled explicitly. A model that just uses the learning mechanisms of the architecture would be a more general one. Therefore I decided to use associations to represent sequence information. It will be interesting to see, how the model exceeds simple chaining models, which have been criticized for being unable to display the properties of memory for sequential order (Brown, 1997).

So far, the model would basically predict associative learning of sequences. In order to model the skipping of steps depending on workload, an additional assumption is necessary. I assume that some of the scans in the learning phase are fragmentary, too. This results in additional associations that deviate from the sequential structure and in lower baselevel values for the skipped steps. The question why steps might be skipped in the learning phase - inadvertently, or as a part of the training - will be discussed in the final section.

A further assumption that had to be determined was how to model workload. A central aspect of workload is the intensive use of working memory. High working memory load results when much information must be kept active (Miyake & Shaw, 1999). In ACT-R, working memory content is defined as the highly activated part of declarative memory. The main process that mediates this activation is the spread of source activation (Lovett, Reder, & Lebiere, 1999). Sources of activation are all chunks in the slots of the current goal. As the total source activation in ACT-R is limited to a constant value *W*, it is divided by the number of chunks in the goal. The more chunks are in the goal, the less source activation each can spread. Put briefly, working memory load varies with the number of chunks that are in the goal. To simulate high workload, I filled some slots of the goal with additional chunks that were not related to the scanning task. These slots were empty in the low workload condition.

The following sections describe the model in more detail[1]. The model's declarative memory contains representations of the six displays (A - F), representations of "look commands" that guide the model to look at the appropriate place in the environment, and representations of the displayed values. Additionally, there are chunks representing "other information" which are used to simulate workload and are otherwise meaningless in the current model.

The chunk type used for the displays has two slots: One for the look command that guides the attention to the display, and one for the value shown by the display. The chunk type for the look commands stores the display that shall be scanned by that command in its only slot. The chunk type used for the goal is defined as follows:

```
(chunk-type main-goal focus step result context d1 d2 d3)
```

The focus slot contains either the look command or the representation of the display that is read. The perceived value is stored in the result slot. The context slot contains a chunk denoting the context. In the simulations presented here, this chunk is a constant, but the slot can be used to simulate different contexts. The slots `d1`, `d2`, and `d3` are used to simulate different workload conditions.

A learning trial begins with the chunk `start` in the step slot of the goal. The production `get-look` asks for a place to look. The look command is then provided and filled into the step slot of the goal. The next production `learn-look` retrieves the memory representation of the command, moves it to the focus slot, and sets the step slot to `nil`. Production `exec-look` then looks at the place stored in the look command. The two following productions process the display and the displayed value, and return the goal with the representation of the display in its focus slot and a command to ask for the next place to look. The production `get-look` then starts a new cycle.

[1] Words denoting ACT-R elements such as chunks and production rules are printed in `courier new`.

The critical step for learning associations that can be used later takes place when the new look command is retrieved while the last display is still in the focus. This is illustrated with the production rules shown in table 1. The rule on the left retrieves the look command `=comm` which is already element of the goal chunk, because it was provided externally. For example, if `=display` is `display-A` and `=comm` is `look-B`, then the association between `display-A` and `look-B` is strengthened when `look-B` is retrieved. Processing in the test phase is very similar to that in the learning phase, except that the next place to look is not provided externally.

----

insert table 1 here

It is very important to note that the retrieval of the look command is not symbolically constrained in the production `retrieve-step`. This means that the most active look command is retrieved from memory. However, it is quite likely to retrieve the look commands in the learned sequence. Suppose for example, that display A has just been processed so that `display-A` is still in the focus. In this situation, `display-A` spreads activation to `look-B` via the associative link learned earlier, raising the probability of retrieving `look-B`.

Simulated Experiment

The influence of the factor workload was investigated in a simulated experiment with 40 simulation runs. The ACT-R parameters were set as follows: baselevel-learning = 0.5 (default), associative-learning = 5.0, activation-noise-s = 0.05, partial-matching = off, all other parameters had their default values. In the low workload condition, the three slots `d1`, `d2`, and `d3` were empty. In the high workload condition, they were filled with chunks. These chunks had been used in the learning phase already, but in a counterbalanced order, so that they had rather low and similar associations with all steps of the sequence.

The environment of the model consisted of four displays A, B, C, D, E, and F. In the learning phase, the model was advised to scan these displays in alphabetical order. However, in one third of the learning trials, the model was advised to scan only a subset of the displays. It was the sequence A-B-D-F. The learning phase comprised 18 complete and 9 reduced sequences in alternate order, with simulated breaks between the trials. In the test phase, the model performed four scans, two for each workload condition. As dependent measures, I classified the resulting scans as shown in figure 1, and counted how often each display was left out in a scan.

----

insert figure 1 here

In the condition with low workload, 82% of the scans were performed completely in the correct order (ABCDEF). This was only true for 28% of the scans in the high workload condition. In that condition, a range of different patterns occurred, most of them characterized by the skipping of one step (ABDEF: 15%, ABCDF: 18%), or two steps (ABDF: 20%). Unexpectedly, there were 20% of longer patterns. Many of these include a jump back to display A somewhere during the scan, some include a jump back to B, and a few contain even two jumps back. There was never more than one step skipped under low workload.

Figure 2 shows the aggregated proportions of which steps were skipped in all scan patterns. Apart from two exceptions, only the steps C and E were skipped, about equally often. This shows that the skipping of steps in the high workload condition is not just due to noise, but closely reflects what has been done in the learning phase.

----

insert figure 2 here

To understand the behavior of the model, it is helpful to have a look at the resulting baselevel and sji values which are indicated by light and bold boxes and lines in figure 3. At the end of the learning phase, there are strong associative links between the displays and the look commands that guide attention to the following display. Weaker associations have been learned between `B` and `look-D`, and `D` and `look-F`, because these transitions occurred less often. The baselevels of `look-C` and `look-E` are lower than those of the other look commands, because they were used less often.

----

insert figure 3 here

Now let us look at the point when display B has just been processed and the model tries to retrieve the next step. `B` spreads activation to `look-C`, but also to `look-D`. Suppose there is no noise. If baselevel activation alone would determine the retrieval of a chunk, `look-D` would always be retrieved in that situation, because its baselevel value is higher than that of `look-C`. If associative activation alone would determine retrieval, `look-C` would always be retrieved, because it is stronger activated by the source `B` than `look-D`. Source spread is lower under conditions of high workload, because source activation is divided between more elements. Therefore, under that condition, the proportion of baselevel activation in the total activation is high enough to raise the total activation of `look-D` slightly above that of `look-C`. Under low workload, the relative influence of source spread is higher, resulting in a higher total activation of `look-C`.

Discussion

The model predicts the selective skipping of steps of a scan pattern depending on workload conditions. This is achieved with very few model specific assumptions. One of them is that in the learning phase not only the whole sequence is practiced, but also a reduced one. Most of the model's behavior arises from the underlying architecture. In essence, it demonstrates the relative influences of two independent learning mechanisms - baselevel learning and associative learning - under different circumstances. As the influence of associative learning diminishes with decreasing source activation, the relative influence of baselevel activation increases. This basic mechanism generally predicts that effects based on associations weaken with increasing workload.

The scanning task modeled here partially resembles serial recall paradigms for which several models have been proposed. These models have to predict a wide range of phenomena which can not be explained by chaining models that only assume the learning of pairwise associations between consecutive items (Brown, 1997). It is important to note that the present model is not a simple chaining model, but in fact, incorporates many of the mechanisms used in the more sophisticated models of serial memory. Again, most of them arise from the inherent properties of the architecture. The finding that memory for the items is independent of memory for the order maps well onto the distinction between baselevel learning and associative learning, with the first process accounting for item learning and the second for sequence learning. Further, as in context-based models (Houghton, 1990), the last step is not the only one that helps to find the next step. Almost naturally in ACT-R models, some context information is stored in the slots of the goal-chunk that adds source activation useful to determine the correct next step.

Some models of serial recall entail the inhibition of the last retrieved item in order to avoid perseveration (Houghton, 1990; Burgess & Hitch, 1992). In the present model, perseveration is prevented by two features. The first is that the critical association is not learned between items of the same type. Since the only constraint for the associative retrieval is the chunk type, using a chunk of a different type as the main retrieval cue impedes the repeated retrieval of the highly activated cue. In my model, the critical positive associations are learned between the display and the next step. Additionally, a negative association between the display and the current step is learned, because the look command guiding the look to a display is never retrieved while the display itself is in the focus (e.g. `look-B` is never retrieved while `display-B` is in the focus).

There is another problem with associations when the same step occurs in several sequences, at each case followed by a different next step. If the associations with the competing steps are equally strong, a chunk that represents the context and spreads source activation to the right step can deliver the decisive information. This mechanism proved to be successful in explorative simulations with the model. It fails, however, in situations where the associations between the current and the competing next steps are highly unequal. In that case, the problem can be solved by using separate chunks for the same step in each context with the command itself bound in a slot.

Finally, let me return to the applied starting-point. If one accepts that the model reflects some basic properties of the learning of sequential action, it follows that students should not be left without guidance unless they are sure to perform the entire sequence of steps correctly. This is because inadvertent step-skipping will contribute to the learning result. Particularly at the beginning of the training, when associations and baselevels have not yet stabilized, erroneous skipping of steps tends to be maintained. If it is important to perform an entire sequence under all conditions, as e.g. in FMC programming, step-skipping in the learning phase involves an inherent danger that may take a lot of training to overcome.

In some cases, however, it might be desirable to skip steps depending on the situation, e.g. to save time. In that case the reduced pattern should be trained beside the complete pattern. In high workload situations, the less trained steps would then be skipped automatically. The emergence of the appropriate pattern can additionally be supported by emphasizing discriminative context cues which will contribute to the retrieval of the right steps by adding decisive activation.

Probably, the described effects are not the only reason why pilots miss pieces of information during busy phases of flight. Another opportunity for making such errors is when temporarily displayed information is not noticed because pilots are preoccupied with other things. However, the virtue of the model is that it points to a less obvious source of potential errors and has clear implications for training.

References

Anderson, J. R. & Lebiere, C. (1998). *Atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum Associates.

Brown, G. D. A. (1997). Formal models of memory for serial order: a review. In: M. A. Conway (Ed.), *Cognitive models of memory*. Cambridge, MA: The MIT Press. (pp.47-77).

Burgess, N. & Hitch, G. J. (1992). Towards a network model of the articulatory loop. *Journal of Memory and Language*, *31,* 429-460.

Houghton, G. (1990). The problem of serial order: a neural network model of sequence learning and recall. In: R. Dale, C. Mellish, & M. Zock (Eds.), *Current research in natural language generation*. London: Academic Press. (pp.287-319).

Boehm-Davis, D. A., Schoppek, W., Diez, M., Hansberger, J.T., Holt, R.W., & Ikomi, P.A. (2000). Cognitive modeling of airline crew automation errors. Poster at the 14th Triennial Congress of the International Ergonomics Association / the 44th Annual Meeting of the Human Factors and Ergonomics Society.

Irving, S., Polson, P., & Irving, J. E. (1994). A GOMS analysis of the advanced automated cockpit. In: Proceedings of the CHI Conference 1994. (pp.344-350).

Lovett, M. C., Reder, L. M., & Lebiere, C. (1999). Modeling working memory in a unified architecture. An ACT-R perspective. In: A. Miyake & P. Shaw (Eds), *Models of working memory. Mechanisms of active maintenance and executive control*. (pp.135-182).

Miyake, A. & Shaw, P. (Eds). (1999). *Models of working memory. Mechanisms of active maintenance and executive control*. Cambridge, UK: Cambridge University Press.

Sarter, N. B. & Woods, D. D. (1992). Pilot interaction with cockpit automation: operational experiences with the flight management system. *The International Journal of Aviation Psychology*, *2*, 303-321.

Sarter, N. B. & Woods, D. D. (1995). How in the world did we ever get into that mode? Mode awareness in supervisory control. *Human Factors, 37,* 5-19.

Wiener, E. L. (1989). *Human factors of advanced technology ("glass cockpit") transport aircraft* (NASA Contractor Rep. No. 177528). Moffett Field, CA: NASA-Ames Research Center.

```
                     (p learn-look               (p retrieve-look
                       =goal>                      =goal>
                          isa learn-goal              isa main-goal
condition                 focus =display              focus =display
  part                    step =comm                  step nil
                       =comm>                      =comm>
                          isa command                  isa command
                      ==>                         ==>
                       =goal>                      =goal>
 action                   focus =comm                  focus =comm
  part                    step nil)                    step nil)
```

Table 1: Production rules that retrieve the look command in the learning phase and in the test phase. In the production `learn-look`, the look-command (`=comm`) is already element of the goal chunk because it was provided externally. This command is then retrieved and moved to the `focus` slot of the goal. In the production `retrieve-look`, the retrieval of the look command is not symbolically constrained.

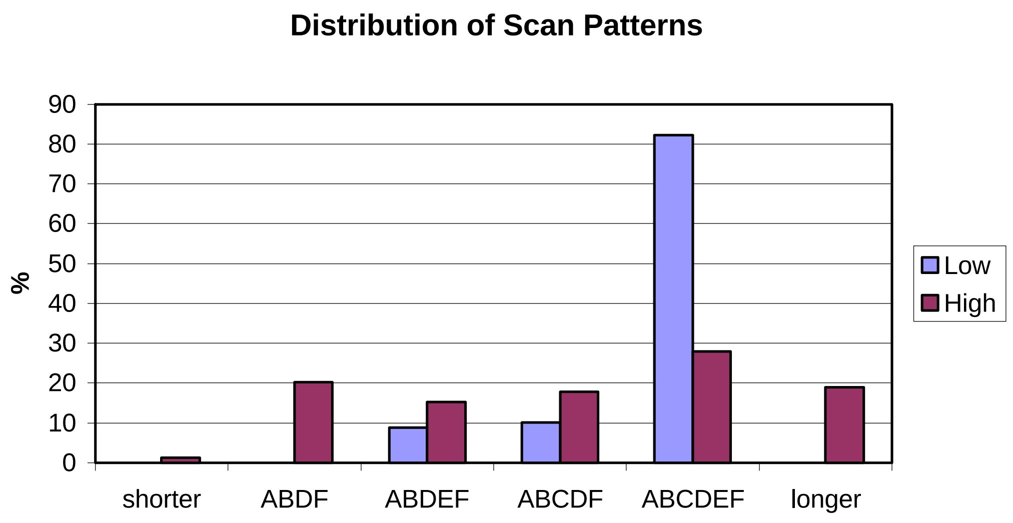

Figure 1: Distribution of the resulting scan patterns under low and high workload. "Shorter" means that the pattern was shorter than four steps, "longer" means that it was longer than six steps.

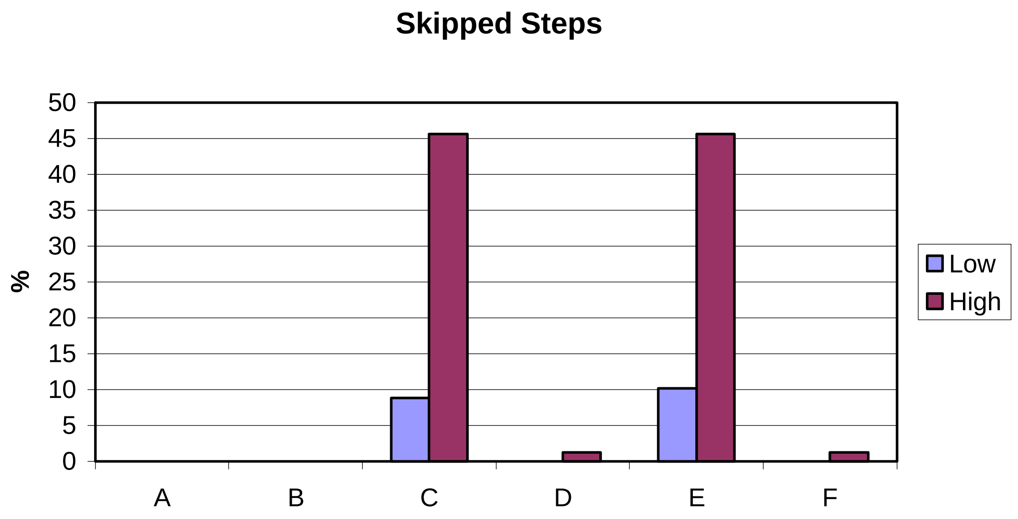

Figure 2: Steps that were skipped under low and high workload. For each step the proportion of scans in which that step was skipped is indicated.

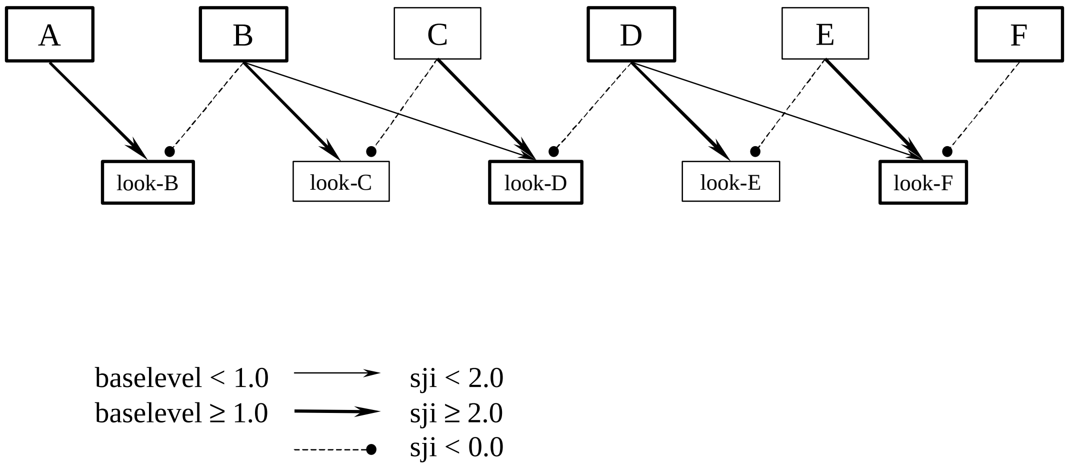

Figure 3: Selected baselevel values and associative weights between representations of displays and look commands at the end of the learning phase. Bold lines or boxes denote higher associations or higher baselevels, dashed lines denote negative associations.